Recover

Introduction

We all make mistakes… Sometimes we delete our files and need it later. There are several third party softwares that can help us to retrieve files. If you haven't done it, at least you must have seen it on movies how forensic experts retrieve files from drives. Have you wondered how it is done?

In this program, we will write code to retrieve **jpeg** and **pdf** files deleted previously.

Background

Thankfully, in the computer world, “deleted” tends not to mean “deleted” so much as “forgotten.” Even though the camera insists that the card is now blank, we’re pretty sure that’s not quite true. Indeed, we’re hoping (er, expecting!) you can write a program that recovers the photos for us!

Even though JPEGs are more complicated than BMPs, JPEGs have “signatures,” patterns of bytes that can distinguish them from other file formats. Specifically, the first three bytes of JPEGs are **0xFF, 0xD8, 0xFF** and from first byte to third byte, left to right. The fourth byte, meanwhile, is either **0xe0, 0xe1, 0xe2, 0xe3, 0xe4, 0xe5, 0xe6, 0xe7, 0xe8, 0xe9, 0xea, 0xeb, 0xec, 0xed, 0xee, or 0xef**. Put another way, the fourth byte’s first four bits are **1110**.

Odds are, if you find this pattern of four bytes on my memory card, they demarcate the start of a JPEG. To be fair, you might encounter these patterns on some disk purely by chance, so data recovery isn’t an exact science.

Fortunately, digital cameras tend to store photographs contiguously on memory cards, whereby each photo is stored immediately after the previously taken photo. Accordingly, the start of a JPEG usually denotes the end of another. However, digital cameras often initialize cards with a FAT file system whose “block size” is 512 bytes (B). The implication is that these cameras only write to those cards in units of 512 B. A photo that’s 1 MB (i.e., 1,048,576 B) thus takes up 1048576 ÷ 512 = 2048 “blocks” on a memory card. But so does a photo that’s, say, one byte smaller (i.e., 1,048,575 B)! The wasted space on disk is called “slack space.” Forensic investigators often look at slack space for remnants of suspicious data.

In case of pdf files, the beginning signature includes **0x25, 0x50, 0x44, 0x46 and 0x2D** as the first five bytes.

Program structure

- Open the drive in read only mode you want to retrieve.
- Each time read 512 Bytes and check the first 4 bytes
    - if they indicate the start of jpeg.
        - Close previous and create a new jpeg file.
        - Write those buffered 512 bytes on a new file just created.
    - if they indicate the start of the pdf.
        - Close previous and create a new pdf file.
        - Write those buffered 512 bytes on a new file just created.
    - If it is not, write those 512 Bytes on the file we are writing.
- Repeat the process until the end of the file(your external drive).
- If the file we are writing is not closed, close the file

Source code can be found here.

Testing

The program needs some permission to run. So, It is recommended to run the code using terminal (on mac) or command prompt (on Windows).

- Go to the root of the .c file you wrote. I have saved it as “recover.c” in documents
    - You can use the “cd” command to get to the root of the file.
- Run the code using command “sudo ./<your_program>” (for me it’s “sudo ./recover.c”)
- You might require to enter your administrator password and it takes time.
- Open the window of the folder where your program is kept. Found jpeg and pdf will be saved in the same folder named serially in 3 digits. For example *000.jpg, 001.pdf, 002.jpg, 003.jpg, 004.pdf* etc